Improving the Performance of High-Energy Physics Analysis through Bitmap Indices

Kurt Stockinger[1,2], Dirk Duellmann[1], Wolfgang Hoschek[1,3], and Erich Schikuta[2]

[1] CERN IT Division, European Organization for Nuclear Research
CH-1211 Geneva, Switzerland
{Kurt.Stockinger, Dirk.Duellmann, Wolfgang.Hoschek}@cern.ch
[2] Institute for Computer Science and Business Informatics
University of Vienna, A-1010 Vienna, Austria
Erich.Schikuta@univie.ac.at
[3] Institute of Applied Computer Science
University of Linz, A-4040 Linz, Austria

Abstract. Bitmap indices are popular multi-dimensional data structures for accessing read-mostly data such as data warehouse (DW) applications, decision support systems (DSS) and on-line analytical processing (OLAP). One of their main strengths is that they provide good performance characteristics for complex adhoc queries and an efficient combination of multiple index dimensions in one query. Considerable research work has been done in the area of finite (and low) attribute cardinalities. However, additional complexity is imposed on the design of bitmap indices for high cardinality or even non-discrete attributes, where different optimisation techniques than the ones proposed so far have to be applied.

In this paper we discuss the design and implementation of bitmap indices for High-Energy Physics (HEP) analysis, where the potential search space consists of hundreds of independent dimensions. A single HEP query typically covers 10 to 100 dimensions out of the whole search space. In this context we evaluated two different bitmap encoding techniques, namely equality encoding and range encoding. For both methods the number of bit slices (or bitmap vectors) per attribute is a central optimisation parameter. The paper presents some (first) results for choosing the optimal number of bit slices for multi-dimensional indices with attributes of different value distribution and query selectivity. We believe that this discussion is not only applicable to HEP but also to DW, DSS and OLAP type problems in general.[1]

1 Introduction

One of the big challenges at CERN (the European Organization for Nuclear Research in Geneva, Switzerland) is the management of the large amount of

[1] 11th International Conference on Database and Expert Systems Applications DEXA 2000, London, Greenwich, UK, September 2000

data and the complexity of objects that are the results of the HEP experiments. In particular, sub-atomic particles are accelerated to nearly the speed of light and then collided. Such collisions are called *events* and are measured at time intervals of only 25 nanoseconds for some of the new experiments. According to [8] around 5 Petabyte of data will be written per year that will be analysed by some 5,000 physicists around the world over a life span of two to three decades.

Currently physics analysis tasks are based on sequentially scanning the preselected event space, obviously not very efficient for queries with small selectivities. In this case the usage of a proper multi-dimensional index data structure accelerates these processes by orders of magnitude.

In the literature on multi-dimensional access methods a variety of indices are proposed [5] ranging from spatial data access methods like the R-tree [6] or the BV-tree [4] and its variants to non-spatial data access methods like the Pyramid-tree [1]. However, all these indices are optimised for transaction processing, i.e. inserts, updates, deletes, etc. what is not the major need of HEP analysis.

Similar to DW applications and DSSs, HEP data are read-mostly and the access methods are characterised by multi-dimensional, highly complex queries. What is more, most of the queries are so called *partial range queries* where only a small subset of the whole search space is accessed. Multi-dimensional access methods like the Pyramid-tree show their best performance characteristics for *full range queries* and are thus only sub-optimal for *partial range queries*.

We therefore propose to use bitmap indices, which are optimised for processing complex adhoc queries in read-mostly environments. The basic idea of a bitmap index is to store one vector of bits per distinct attribute value (e.g. possible attribute values are *colours*). Each bit of the value is mapped to a record. The associated bit is set if and only if the record's value fulfils the property in focus (e.g. the respective value of the record is equal to *red*). [2] [3] [14] studied different kinds of bitmap encoding techniques but only for discrete values. However, additional complexity is imposed on the design and implementation of bitmap indices for non-discrete values since different optimisation techniques to the ones proposed so far have to be applied.

In this paper we make the following contributions to the research on bitmap indices:

- Applying bitmap indices to high performance physics experiments.
  We give a proof of concept that traditional physics analysis can be considerably improved by bitmap indices (BMIs) and show that this technique is very efficient for HEP-specific data distributions. What is more, we introduce *partitioned equality encoding*, which is a variant of equality encoded bitmap indices as used in [12]. In this case one of the most crucial points is the choice of the correct number of bins which highly depends on the number of indexed attributes, i.e. the number of dimensions of our search space.
- Employment and analysis of BMIs on top of an object database managment system (ODBMS).
  In contrast to [12] where bitmap indices are implemented on top of a mass storage system, we present a first implementation on top of an ODBMS,

namely Objectivity/DB [9] . In addition, we also provide an extensive performance analysis and thus characterise the features of bitmap indices for HEP, which can be, without loss of generality, directly applied for any read-mostly environment like DWs and DSSs.

– Studying the impact of non-discrete data values.
  Finally, we discuss the impact of *partitioned range encoding*, which is a new variant of range encoded bitmap indices [2] and is not covered in the research community so far.

The paper proceeds as follows. In Section 2 we give a survey of related work and outline the differences to our approach. In Section 3 and 4 we discuss BMIs for HEP analysis and how we implemented them on top of an ODBMS. A detailed evaluation of our index is presented in Section 5 where we elaborate on the optimal binning of the BMI and apply these results to data distributions which are very common in HEP analysis. In Section 6 we discuss the impact of range encoded BMIs and finally conclude our work in Section 7.

2 Survey and Discussion of Related Work

A detailed discussion on designing bitmap indices based on different encoding schemes is presented in [2] and [3]. In particular, space and time complexities for so-called equality encoded, range encoded and interval encoded bitmap indices are evaluated. Equality encoding (Table. 1 (b)) can be regarded as the most fundamental method that consists of $|A|$ bitmaps (bitmap vectors) where $|A|$ is the cardinality of the attribute to be indexed on. This type of index is optimal for exact match queries of the form $Q_e : v = a_i$. One sided-range queries like $Q_{1r} : v_1 \text{op } a_i$ where $op \in \{<, \leq, >, \geq\}$ show the best performance characteristics with range encoded bitmap indices (Table 2 (b)), which only consist of $|A| - 1$ bitmap vectors. Finally, interval encoding (Table 2 (c)) consists of $\frac{|A|}{2}$ bitmap vectors only and is optimal for two-sided range queries $Q_{2r} : v_1 \text{op } a_i \text{ op } v_2$ where $op \in \{<, \leq, >, \geq\}$. All the optimisation methods presented in their work address discrete attribute values only. However, since our data are contiguous and thus do not have finite cardinalities, different optimisation techniques than the ones proposed so far have to be applied.

Table 1 and 2 depict these different encoding techniques for the same set of attribute values. According to the terminology of [10] [11] the values in Table 1 (a) are referred to as *projection index* whereas the other methods are called *bit sliced* indices.

One of the major problems of simple bitmap indices, namely handling of large cardinality domains, is solved in [15] by range-based indices. A bitmap vector is used to represent a range instead of a distinct value and the entire ranges are partitioned into equally spaced *buckets.* As is the case with the approach described in this paper, range-based indices require additional query processing time to examine the details of all the records in the matched buckets. However, a detailed analysis and a possible solution to the problem of the additional

Table 1. a) Projection Index π_A and b) Equality Encoding E^i

	π_A(R)	E^9	E^8	E^7	E^6	E^5	E^4	E^3	E^2	E^1	E^0
1	4	0	0	0	0	0	1	0	0	0	0
2	3	0	0	0	0	0	0	1	0	0	0
3	4	0	0	0	0	0	1	0	0	0	0
4	5	0	0	0	0	1	0	0	0	0	0
5	7	0	0	1	0	0	0	0	0	0	0
6	8	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	1
9	4	0	0	0	0	0	1	0	0	0	0
10	8	0	1	0	0	0	0	0	0	0	0
11	6	0	0	0	1	0	0	0	0	0	0
12	7	0	0	1	0	0	0	0	0	0	0

overhead for retrieving data from disk ("sieving out" the matching attribute values), was still left an open issue.

Table 2. a) Projection Index π_A, b) Range Encoding R^i and c) Interval Encoding I^i

	π_A(R)	R^8	R^7	R^6	R^5	R^4	R^3	R^2	R^1	R^0	I^4	I^3	I^2	I^1	I^0
1	4	1	1	1	1	1	0	0	0	0	1	1	1	1	1
2	3	1	1	1	1	1	1	0	0	0	0	1	1	1	1
3	4	1	1	1	1	1	0	0	0	0	1	1	1	1	1
4	5	1	1	1	1	0	0	0	0	0	1	1	1	1	0
5	7	1	1	0	0	0	0	0	0	0	1	1	0	0	0
6	8	1	0	0	0	0	0	0	0	0	1	0	0	0	0
7	0	1	1	1	1	1	1	1	1	1	0	0	0	0	1
8	0	1	1	1	1	1	1	1	1	1	0	0	0	0	1
9	4	1	1	1	1	1	0	0	0	0	1	1	1	1	1
10	8	1	0	0	0	0	0	0	0	0	1	0	0	0	0
11	6	1	1	1	0	0	0	0	0	0	1	1	1	0	0
12	7	1	1	0	0	0	0	0	0	0	1	1	0	0	0

Another encoding technique based on binary encoding is proposed by [13] where an attribute value is represented in binary form with only $\lceil log_2 |A| \rceil$ bitmaps. Obviously, the storage overhead is much less for high cardinality attributes when compared to equality encoding or range encoding but even according to the authors, an optimal solution for evaluating the queries might not always exist.

Static and dynamic query optimisation for *continuous range selections* (i.e. one-sided and two-sided range queries) and *discrete range selections* (i.e. queries of the form $v \in a$ and $v \notin a$ are presented in [14]. Static query optimisations are questions concerning the optimal design of bitmaps and algorithms based on logical reductions. Dynamic query optimisation tries to answer questions on inclusion and exclusion for bit-sliced and encoded bitmap indices.

Currently the only work on BMIs for HEP is presented in [12]. Their work is based on a hybrid approach of equality encoded [2] and range-based BMIs [15] on top of a mass storage system. They also use bitmaps for their query optimiser to provide a quick estimate of the size of the requested data. However, an answer to the optimisation problem of the central optimisation parameters for designing a BMI for HEP, namely the optimal number of buckets (or bit slices), was not given yet.

3 Bitmap Indices for HEP

The typical query profile of physicists who wish to retrieve data for their analyses can be regarded as partial range queries, i.e. queries that do not cover all dimensions of the whole search space and thus only a subset of all dimensions of the data is retrieved. What is more, data are read-mostly and skewed.

In our prototype implementation we created a bitmap index for HEP data comprising 10^6 objects, i.e. 1 million events with up to 20 independent attributes. This can be regarded as an index table with a length of 10^6 and a width of 20. We assume that the order of the objects, that are stored in the index, does not change.

Similar to [12] we also use a hybrid approach of equality encoded [2] and range-based BMIs that we call *partitioned equality encoding* or short *equality encoding*. The properties or attributes are partitioned into bins, for example the attribute energy can be binned into several ranges like [0;20) GeV (Giga electron Volt), [20;40) GeV, etc. Afterwards, a bit slice is assigned to each bin, whereas *1* means that the value for the particular event falls into this bin and *0* otherwise.

The steps for performing a two-sided range query of the form $Q_{2r} : v_1 \text{op } a_i$ op v_2 where $op \in \{<, \leq, >, \geq\}$ are as follows. First, the query range has to be interpreted in terms of bins. Thus, we can easily compute how many bins need to be scanned for answering our query. Since each bin represents a range rather than a distinct value, the edge bins are so called "critical" bins, that might only be partially covered by the query condition. In order to "sieve" out the correct events from the *candidate slices*, we need to fetch the event data from disk and check the attribute value against the query condition. We refer to this as the "candidate check overhead" that makes the index highly I/O bound for a large number of candidates in the two candidate slices. Those slices that are covered 100% by the query range, are called *hit slices*. In this case all events that are represented by this slice are hits and do not need any additional checking. A typical example of a two-sided range query with 2 candidate slices and 1 hit slice is depicted in Fig. 1.

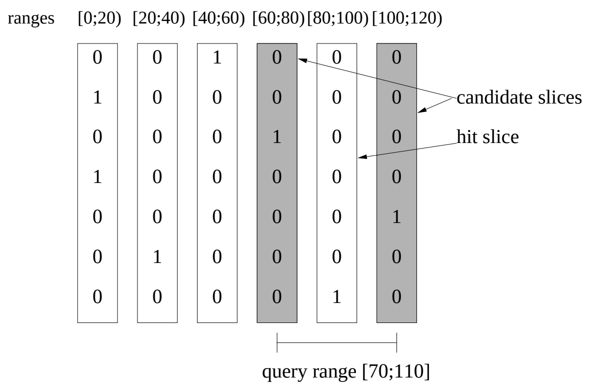

Fig. 1. Two sided-range query on a partitioned equality encoded HEP-BMI

4 Implementation on Objectivity/DB

Basis for our implementation is Objectivity/DB, which is a distributed object database management system for high performance, high availability multi-tier applications. Objectivity/DB provides a robust, scalable multi-threaded database engine. Both the event data and the BMI are implemented in separate databases under one federation which in turn is the highest level of abstraction in Objectivity/DB and allows to be accessed physically distributed databases. From the point of view of the programmer, the whole database system is one logical unit. The main architectural aspects are depicted in Fig. 2.

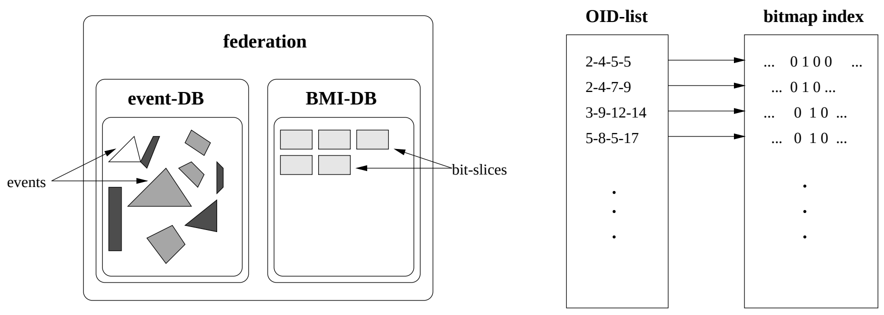

Fig. 2. Architectural overview of the BMI on top of Objectivity/DB

Any object in Objectivity/DB can be directly accessed by its object identifier (OID) which we use for keeping track of the event data. In particular, each physics event is stored as an object and can thus be directly accessed via its OID. This step is necessary, for example, for checking the *candidate slices*.

As we can see on the right side of Fig. 2, one OID-list is maintained in addition to the BMI. For instance, if we want to check the event at position x, we simply refer to the OID list at position x and fetch the event from disk for checking the attribute value against the query condition.

5 Justification of the BMI Approach

We carried out our benchmarks on a Pentium II 400 under Linux Red Hat 5.1. The BMI is implemented on top of Objectivity/DB version 5.1.2. Throughout the rest of this paper all experiments operate on 10^6 events.

5.1 Optimal Binning - Space/Time Complexity

The right number of bit slices (bins) can be regarded as one of the "key parameter" of this kind of bitmap index. A detailed discussion on the behaviour of the BMI with a different number of bins and a different number of indexed attributes is vital for the understanding of BMIs for HEP. To the best knowledge of the authors, this kind of investigation has not been done before for the special needs of HEP data and DW applications in general. The main motivation was a similar implementation of BMIs presented in [12] where 20 bins were chosen. However, no analysis or justification for this "key parameter" is given.

We will first elaborate on the optimal number of bins for queries against 1 indexed attribute and later extend our analysis onto 2, 10 and 20 attributes - that can be summarised as the "most characteristic" use cases of end-user physics analysis in HEP. For simplicity we base our analysis on uniformly distributed data since this gives us the best insight into the performance of the index. Again the domain selectivity $\sigma_{dom} = 100\%$.

As we can see from Fig.3, the optimal binning highly depends on the number of dimensions, that are covered by the range query. Generally speaking we can conclude that the higher the search space, that is covered by the query, the higher is the number of bins for an optimal query performance.

Let us first analyse the graph for queries over one dimension. As we might expect, the performance is worse for 2 bins since both of them are candidates and hence need to be checked against the query constraint. In this case, all events must be fetched from disk. By increasing the number of bins, the number of candidates in the candidate slices gets lower and thus the I/O spent on fetching events from disk is reduced. However, at the same time a larger number of bit slices has to be scanned for performing the Boolean operations on them. The optimal number of bins can be found at that point where these two effects offset each other.

The same effect can be found for higher dimensional queries with the slight difference that the optimum moves to the right, i.e. higher number of bins, due to the higher number of candidates in the candidate slices for a higher number of bins. Obviously, for 2 bins the number of candidates is the same for all dimensions but this number decreases more slowly as the number of dimensions and bins increases.

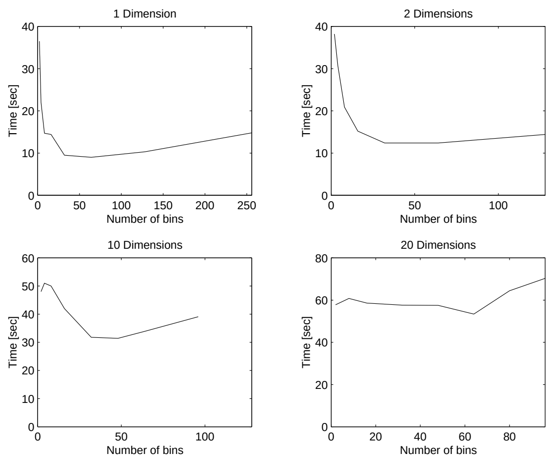

Fig. 3. Optimal binning for queries over various dimensions

5.2 Different Distributions of Physics Data

After studying the behaviour of the BMI on event data following a uniform distribution, we will now motivate our implementation by applying it to typical data distributions that can be found in HEP. During this section we demonstrate the advantage of our index data structure for HEP analysis over conventional methods.

A great majority of physics data follows a Gauss or exponential distribution. Driven by the needs of physics analysis, we studied the behaviour of our BMI on a Gauss distribution with the parameters $\mu = 0$ and $\sigma = 1$, and on an exponential distribution with the parameter $\lambda = 1$. The number of bins is set to 32 and the number of dimensions covered by the query is 10. The results are summarised in Fig. 4

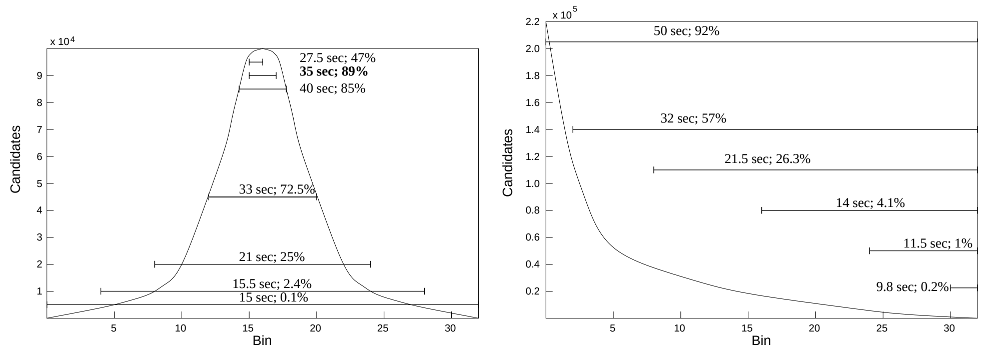

Fig. 4. Variable selectivities - Gauss and exponential distribution

In both figures a schematic view of the underlying data distribution is given. The horizontal bars indicate the domain selectivity of the query whereas the percentage on the bars refers to the selectivity of the candidate objects. In addition, the time for executing the query is given. As we can see in the figures the query time highly depends on the selectivity of the candidate objects and not on the domain selectivity.

The performance of the BMI is highest for the “lower end” of highly skewed data (e.g. the right most events that follow an exponential distribution). However, we also have to point out the poor performance of range queries at the “higher end” (worst case).

One possible optimisation for reducing the negative effects of the worst case would be a dynamic binning that adopts to the distribution of the event data. Since in most cases the access patterns of physics analysis are characterised by range queries around the “lower end” of the event data, our current implementation proofed to be the most promising approach for the HEP experiments at CERN.

6 Partitioned Range Encoding

Since one-sided range queries (Q_{1r}) are the most common kind of query in HEP, we also analysed range encoding [2] that performs considerably better than equality encoding. In contrast to [2] who based their optimisation techniques on discrete attribute values (where the problem of candidate and hit slices does not occur), we apply this method to contiguous values and thus a new optimisation method has to be considered. We refer to our approach as *partitioned range encoding* or in short *range encoding*.

The main advantage over equality encoded BMIs is that in the worst case only one bit slice has to be scanned for one-sided range queries per dimension (independent of the selectivity of the query). As for equality encoded bitmaps, all bit slices have to be scanned.

Since we have already studied the behaviour of equality encoded BMIs and raised the I/O problem of candidate slices, we can easily conclude from these observations on the impact of range encoding. As already mentioned, in the worst case one bit slice needs to be scanned for one-sided range queries per dimension. This also implies that we have to consider only one *candidate slice* and no *hit slice* at all.

Let us analyse the performance characteristics of one-sided range queries with both equality encoded and range encoded BMIs with variable selectivities, 10^6 objects and 10 indexed attributes. Again we studied the performance characteristics of the BMI on uniformly distributed data.

As we can see in Fig. 5, the query time for equality encoded BMIs increases with increasing selectivities (i.e. a higher number of *hit slices* has to be read) whereas the query time for range encoded BMIs is more or less constant for all selectivities (since no *hit slices* at all have to be scanned).

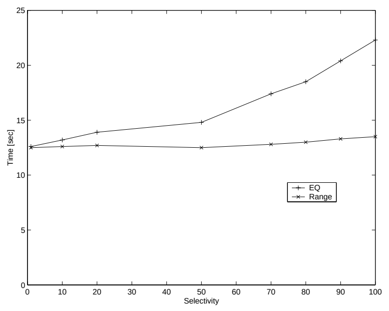

Fig. 5. Partitioned Equality Encoding (EQ) vs. Partitioned Range Encoding (Range)

The fact that only one bit slice needs to be scanned gives us much more freedom in extending the number of bins until the theoretical maximum of the cardinality of the attribute value. Since we are dealing with contiguous values, we do not have a finite cardinality and hence are mainly "restricted" by the space complexity. We thus end up in a "practical" optimisation problem, that is constraint by the available disk space.

Experimentally we can show that the behaviour of a range encoded BMI both for Q_{1r} and Q_{2r} corresponds to a partial scan over the event data. The performance characteristics are thus highly dependent on the characteristics of the underlying OODBMS, namely Objectivity/DB and the disk. In particular, for page selectivities between 5% and 100%, the read rate is almost linear. However, a significant speedup is achieved, if the page selectivity is smaller than 5% [7] which in turn is very common for multi-dimensional range queries in HEP analysis.

7 Conclusion

We have given performance studies of BMI for HEP data and pointed out the main difference to other studies on BMI - those that concentrated on only discrete attribute values. The main bottleneck has been shown to be the checking of the candidate slices due to the additional I/O for fetching the event data from disk in addition to the I/O for the BMI.

We have designed and implemented our BMI on top of a commercial object database management system, namely Objectivity/DB and used different bitmap encoding techniques and different data distributions for our analysis. As for *partitioned equality encoding* and uniformly distributed data we solved the "candidate-bottleneck" by increasing the number of bins and came to an optimal query performance. This optimum can be regarded as a trade-off between a high number of candidates and consequently more I/O on the event data vs. a low number of candidates and therefore a higher number of bins.

Since HEP queries are mainly one-sided range queries, we also studied *partitioned range encoding* where we discussed a completely new problem, namely the behaviour of the BMI on attributes with infinite cardinality (as it is true for non-discrete values). We showed that the performance of range encoded BMIs clearly outperforms equality encoded BMI. However, there is no "optimal" number of bins, as is the case for equality encoded BMI. This gives the designer of BMI for HEP data a high degree of freedom since the number of bins (and indirectly the number of candidates) and thus the performance of the entire index is "only" restricted by the capacity of the storage space.

References

1. S. Berchtold, C. Boehm, H.-P. Kriegel, The Pyramid-Tree: Breaking the Curse of Dimensionality, SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 1998, Seattle, Washington, USA
2. C. Chan, Y.E. Ioannidis, Bitmap Index Design and Evaluation, SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 1998, Seattle, Washington, USA
3. C. Chan, Y.E. Ioannidis, An Efficient Bitmap Encoding Scheme for Selection Queries, SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1999, Philadephia, Pennsylvania, USA
4. M. Freestone, A General Solution of the n-dimensional B-tree Problem, SIGMOD Record (ACM Special Interest Group on Management of Data 24(2), June 1995
5. V. Gaeda, O. Guenther, Multidimensional Access Methods, Computing Surveys 30, September 1998
6. A. Guttman, R-trees: A Dynamic Index Structure for Spatial Searching, Proc. ACM SIGMOD, Int. Conf. on Management of Data, 1984
7. K. Holtman, P. v. d. Stok, I. Willers, Automatic Reclustering Objects in Very Large Databases for High Energy Physics, Proceedings of IDEAS 1998, Cardiff, UK
8. http://www.cern.ch/MONARC/
9. http://www.objectivity.com/
10. P. O'Neil, Informix and Indexing Support for Data Warehouses, Database and Programming Design, February 1997
11. P. O'Neil, D. Quass, Improved Query Performance with Variant Indexes, Proceedings ACM SIGMOD International Conference on Management of Data, May 1997, Tucson, Arizona, USA
12. A. Shoshani, L.M. Bernardo, H. Nordberg, D. Rotem, A. Sim, Multidimensional Indexing and Query Coordination for Tertiary Storage Management, 11th International Conference on Scientific and Statistical Database Management, Proceedings, Cleveland, Ohio, USA, July 1999
13. M. Wu, A.P. Buchmann, Encoded Bitmap Indexing for Data Warehouses, Proceedings of the Fourteenth International Conference on Data Engineering, February 1998, Orlando, Florida, USA
14. M. Wu, Query Optimization for Selections Using Bitmaps, SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1999, Philadephia, Pennsylvania, USA
15. K. Wu, P.S. Yu, Range-Based Bitmap Indexing for High-Cardinality Attributes with Skew, Technical Report, IBM Watson Research Center, May 1996